IN THE CLAIMS

1. (currently amended) A method, comprising:

inserting an on-processor register allocation instruction within a machine code description of a routine if a function call instruction is found within said routine, said inserted on processor register allocation instruction to allocate less on-processor register space for the use of said routine than an amount of on-processor register space allocated for the use of said routine by another on-processor register allocation instruction that is executed prior to said inserted on processor register allocation instruction.

2. (original) The method of claim 1 further comprising configuring said allocation instruction to allocate only for the live information that exists within said routine when said inserted allocation instruction is executed.

3. (original) The method of claim 2 wherein said live information is determined by identifying information that is referred to before and after said function call.

4. (original) The method of claim 3 wherein said information identified after said function call extends to an exit block of said routine.

5. (original) The method of claim 4 wherein the worst case path to said exit block is allocated for.

6. (original) The method of claim 3 wherein said information identified after said function call extends to a post-dominator block of said routine.

7. (original) The method of claim 6 wherein the worst case path to said post-dominator block is allocated for.

8. (original) The method of claim 2 wherein said live information is information that is local to said routine.

9. (previously presented) The method of claim 8 wherein a processor said routine is to be executed upon has its associated register space partitioned into register space used only for local information and register space used only for global information, said allocation instruction pertaining only to said register space used for local information.

C1

10. (original) The method of claim 2 wherein said live information includes global information.

11. (previously presented) The method of claim 1 wherein said allocation instruction is inserted just before a machine code representation of said function call.

12. (original) The method of claim 1 wherein said allocation instruction is inserted in a pre-dominator basic block of said function call.

13. (original) The method of claim 12 wherein said allocation instruction is inserted in said pre-dominator basic block of said function call if there exists a post-dominator basic block of said function call.

14. (currently amended) A method comprising:

inserting an on-processor register allocation instructions within a machine code description of a routine because a functional characteristics selected from the group consisting of:

a) a loop that exists within a control flow graph of said routine;

b) a software pipelined loop; and,

c) a function call to another routine;

~~are~~ is discovered within said routine, said inserted on-processor register allocation instruction to allocate on processor register space for the use of said routine.

15. (previously presented) The method of claim 14 wherein at least one of said discovered functional characteristics includes said loop that exists within a control flow graph of said routine.

16. (previously presented) The method of claim 15 wherein the allocation instruction inserted for said loop is inserted above said loop in said control flow graph.

17. (original) The method of claim 16 wherein said allocation instruction allocates for a worst case path to an exit block of said routine.

18. (original) The method of claim 16 wherein said allocation instruction allocates for a worst case path to a post-dominator block of said routine.

19. (previously presented) The method of claim 14 wherein at least one of said discovered functional characteristics includes said software pipelined loop.

20. (original) The method of claim 19 wherein the allocation instruction inserted for said software pipelined loop is inserted above said loop in said control flow graph.

C1

21. (previously presented) The method of claim 20 wherein said allocation instruction allocates for a worst case path to an exit block of said routine.

22. (original) The method of claim 21 wherein said allocation instruction allocates for a worst case path to a post-dominator block of said routine.

23. (previously presented) The method of claim 14 wherein said one or more functional characteristics include a function call.

24. (previously presented) The method of claim 14 further comprising determining the number of on-processor registers to be allocated for an allocation instruction after a functional characteristic is found.

25. (previously presented) The method of claim 24 wherein all functional characteristics from said group within said routine are discovered before said determining is performed.

26. (original) The method of claim 24 wherein said determining is performed before a next functional characteristic is discovered.

27. (original) The method of claim 14 further comprising building an understanding of said routine's control flow graph before said searching is performed.

28. (currently amended) A method, comprising:

~~a)~~ performing a first allocation for a first amount of on-processor register space at the entry block of a routine, said first amount of on-processor register space for the use of said routine;

~~b)~~ performing a second allocation for a second amount of on-processor register space for the storage of live information ~~within~~ of said routine ~~at the time of said second allocation~~ when said routine performs a function call to a second routine, said second amount less than said first amount;

~~c)~~ performing said function call to said second routine; and,

~~d)~~ performing a third allocation for a third amount of on-processor register space at the entry block of said second routine, said third amount of allocated on-processor register space ~~overlapping with~~ for the use of said second routine and including a region of said first amount of allocated on-processor register space

that stores stale information of said routine when said routine said performs said function call to said second routine.

29. (original) The method of claim 28 wherein said live information is determined by identifying information that is referred to before and after said function call.

30. (original) The method of claim 29 wherein said information identified after said function call extends to an exit block of said routine.

31. (original) The method of claim 30 wherein the worst case path to said exit block is allocated for.

32. (original) The method of claim 29 wherein said information identified after said function call extends to a post-dominator block of said routine.

33. (original) The method of claim 32 wherein the worst case path to said post-dominator block is allocated for.

34. (original) The method of claim 28 wherein said live information is information that is local to said routine.

35. (previously presented) The method of claim 34 wherein a processor said routine is to be executed upon has its associated register space partitioned into register space used only for local information and register space used only for global information, said allocation instruction pertaining only to said register space used for local information.

36. (original) The method of claim 28 wherein said live information includes global information.

37. (original) The method of claim 28 wherein said second allocation is performed just before said function call.

38. (original) The method of claim 28 wherein said second allocation is performed in a pre-dominator basic block of said function call.

39. (currently amended) The method of claim ~~28~~38 wherein said second allocation is performed in said pre-dominator basic block of said function call if there exists a post-dominator basic block of said function call.

40. (original) The method of claim 28 further comprising compiling said routine.

41. (currently amended) A machine readable medium having stored thereon sequences of instructions which are executable by a digital processing system, and which, when executed by the digital processing system, cause the system to perform a method comprising:

inserting an on-processor register allocation instruction within a machine code version of a routine if a function call instruction is found within said routine, said inserted on processor register allocation instruction to allocate less on-processor register space for the use of said routine than an amount of on-processor register space allocated for the use of said

routine by another on-processor register allocation instruction that is executed prior to said inserted on processor register allocation instruction.

42. (original) The machine readable medium of claim 41 further comprising instructions which cause a processor that executes said routine to configure said allocation instruction to allocate only for the live information that exists within said routine when said inserted allocation instruction is executed.

43. (original) The machine readable medium of claim 42 wherein said live information is determined by identifying information that is referred to before and after said function call.

44. (original) The machine readable medium of claim 43 wherein said information identified after said function call extends to an exit block of said routine.

45. (original) The machine readable medium of claim 44 wherein the worst case path to said exit block is allocated for.

46. (original) The machine readable medium of claim 43 wherein said information identified after said function call extends to a post-dominator block of said routine.

47. (original) The machine readable medium of claim 46 wherein the worst case path to said post-dominator block is allocated for.

48. (original) The machine readable medium of claim 42 wherein said live information is information that is local to said routine.

49. (previously presented) The machine readable medium of claim 48 wherein said processor said routine is to be executed upon has its associated register space partitioned into register space used only for local information and register space used only for global information, said allocation instruction pertaining only to said register space used for local information.

50. (original) The machine readable medium of claim 42 wherein said live information includes global information.

C1

51. (previously presented) The machine readable medium of claim 41 wherein said allocation instruction is inserted just before a machine code representation of said function call.

52. (original) The machine readable medium of claim 41 wherein said allocation instruction is inserted in a pre-dominator basic block of said function call.

53. (original) The machine readable medium of claim 52 wherein said allocation instruction is inserted in said pre-dominator basic block of said function call if there exists a post-dominator basic block of said function call.

54. (currently amended) A machine readable medium having stored thereon sequences of instructions which are executable by a digital processing system,

and which, when executed by the digital processing system, cause the system to perform a method comprising:

inserting an on-processor register allocation instructions within a machine code description of a routine because a functional characteristics selected from the group consisting of:

a) a loop that exists within a control flow graph of said routine;

b) a software pipelined loop; and,

c) a function call to another routine;

~~are~~ is discovered within said routine, said inserted on-processor register allocation instruction to allocate on processor register space for the use of said routine.

55. (previously presented) The machine readable medium of claim 54 wherein at least one of said discovered functional characteristics includes said loop that exists within a control flow graph of said routine.

56. (previously presented) The machine readable medium of claim 55 wherein the allocation instruction inserted for said loop is inserted above said loop in said control flow graph.

57. (original) The machine readable medium of claim 56 wherein said allocation instruction allocates for a worst case path to an exit block of said routine.

58. (original) The machine readable medium of claim 56 wherein said allocation instruction allocates for a worst case path to a post-dominator block of said routine.

59. (previously presented) The machine readable medium of claim 54 wherein at least one of said discovered functional characteristics includes said software pipelined loop.

60. (previously presented) The machine readable medium of claim 59 wherein the allocation instruction inserted for said software pipelined loop is inserted above said loop in said control flow graph.

C1

61. (original) The machine readable medium of claim 60 wherein said allocation instruction allocates for a worst case path to an exit block of said routine.

62. (original) The machine readable medium of claim 61 wherein said allocation instruction allocates for a worst case path to a post-dominator block of said routine.

63. (previously presented) The machine readable medium of claim 54 wherein said one or more functional characteristics include a function call.

64. (previously presented) The machine readable medium of claim 54 further comprising sequences of instructions which cause the system to determine the number of on-processor registers to be allocated for an allocation instruction after a functional characteristic is found.

65. (previously presented) The machine readable medium of claim 64 wherein functional characteristics from said group within said routine are discovered before said determining is performed.

66. (original) The machine readable medium of claim 64 wherein said determining is performed before a next functional characteristic is discovered.

67. (original) The machine readable medium of claim 54 further comprising sequences of instructions which cause the system to build an understanding of said routine's control flow graph before said searching is performed.

68. (currently amended) A machine readable medium having stored thereon sequences of instructions which are executable by a digital processing system, and which, when executed by the digital processing system, cause the system to perform a method, comprising:

~~a)~~ performing a first allocation for a first amount of on-processor register space at the entry block of a routine, said first amount of on-processor register space for the use of said routine;

~~b)~~ performing a second allocation for a second amount of on-processor register space for the storage of live information ~~within~~ of said routine ~~at the time of said second allocation~~ when said routine performs a function call to a second routine, said second amount less than said first amount;

~~c)~~ performing said function call to said second routine; and,

~~d)~~ performing a third allocation for a third amount of on-processor register space at the entry block of said second routine, said third amount of allocated on-

processor register space ~~overlapping with~~ for the use of said second routine and including a region of said first amount of allocated on-processor register space that stores stale information of said routine when said routine said performs said function call to said second routine.

69. (previously presented) The machine readable medium of claim 68 wherein said live information is determined by identifying information that is referred to before and after said function call.

70. (previously presented) The machine readable medium of claim 69 wherein said information identified after said function call extends to an exit block of said routine.

71. (previously presented) The machine readable medium of claim 70 wherein the worst case path to said exit block is allocated for.

72. (previously presented) The machine readable medium of claim 69 wherein said information identified after said function call extends to a post-dominator block of said routine.

73. (previously presented) The machine readable medium of claim 72 wherein the worst case path to said post-dominator block is allocated for.

74. (previously presented) The machine readable medium of claim 68 wherein said live information is information that is local to said routine.

75. (previously presented) The machine readable medium of claim 74 wherein a processor said routine is to be executed upon has its associated register space partitioned into register space used only for local information and register space used only for global information, said allocation instruction pertaining only to said register space used for local information.

76. (previously presented) The machine readable medium of claim 68 wherein said live information includes global information.

77. (previously presented) The machine readable medium of claim 68 wherein said second allocation is performed just before said function call.

78. (previously presented) The machine readable medium of claim 68 wherein said second allocation is performed in a pre-dominator basic block of said function call.

79. (previously presented) The machine readable medium of claim 68 wherein said second allocation is performed in said pre-dominator basic block of said function call if there exists a post-dominator basic block of said function call.

80. (previously presented) The machine readable medium of claim 68 further comprising compiling said routine.

COMMENTS

The enclosed is responsive to the Examiner's Office Action mailed on November 20, 2003. At the time the Examiner mailed the Office Action claims 1 through 80 were pending. By way of the present response, the applicants have: 1) amended claims 1, 14, 28, 39, 41, 54, 68; and, 2) neither added nor canceled any claims. As such, claims 1 through 80 remain pending. The Applicants respectfully requests reconsideration of the present application and the allowance of claims 1 through 80.

Independent Claim 1

Independent claim 1 presently stands rejected under 35 USC 102(b) as being anticipated by U.S. Patent No. 6,230,317 (hereinafter "Wu") and stands rejected under 35 USC 103(a) as being obvious in light of the combination of U.S. Patent 5,367,684 (hereinafter, "Smith") and U.S. Patent No. 5,748,963 (hereinafter, "Orr"). Independent claim 1 presently recites (emphasis added):

> 1. A method, comprising:
> inserting an on-processor register allocation instruction within a machine code description of a routine if a function call instruction is found within said routine, said inserted on processor register allocation instruction to allocate less on-processor register space for the use of said routine than an amount of on-processor register space allocated for the use of said routine by another on-processor register allocation instruction that is executed prior to said inserted on processor register allocation instruction.

Claim 1 recites a pair of allocation instructions ("an on-processor register allocation instruction" and "another on-processor register allocation instruction") that allocate registers for the use of the same

routine. Moreover, claim 1 recites that the later executed allocation instruction allocates for less register space than the earlier executed allocated allocation instruction.

These claimed features stem at least from the Applicants' discussion of Figures 5 and 6 of the present application. That is, Figures 5 and 6 of the present application and their surrounding discussion disclose a technique where an allocation instruction that allocates register space (e.g., space 610 in Figure 6) for a caller routine (as a consequence of the caller routine having a function call) can be configured to allocate for less register space than a previously executed allocation instruction allocated (e.g., space 605 of Figure 6) for the same caller routine (e.g., located at the caller routine's entry block).

By contrast the Wu reference at best only discloses, teaches or suggests an allocation instruction for the called routine rather than the caller routine. As such, the Wu reference fails to cover the Applicants' claimed subject matter. The Wu reference teaches the implementation of a software pipelined loop as a called sub function and the corresponding allocation of register space for the called sub function. See, Wu Col. 3, lines 26-29 and Col.4 lines 5-21. Nothing is said in Wu with respect to the insertion of an allocation instruction that allocates register space for the caller function, however. Therefore the Wu reference fails to anticipate each and every limitation of independent claim 1 of the present application.

Similarly, the combination of Orr and Smith fail to disclose, teach or suggest a causal relationship between the presence of a function call in a caller routine and the insertion of an additional allocation instruction in the caller routine that allocates register space for the caller routine. As discussed at length in the Office Action response mailed October 15, 2003, Orr teaches a wholly irrelevant causal relationship between the presence of a function call instruction and the replacement of function call parameters with a dictionary entry.

Smith teaches a register allocation method (specifically, the allocation of register space to two "register candidates" that are live within the same basic block of instructions through an improved "register candidate usage matrix". See, Smith, Col. 2, lines 5-32; Col.2, line 60 – Col. 3, line 2) but says nothing about the insertion of an allocation instruction. That is, Smith teaches a method that can perhaps be used to determine "how many" registers are to be allocated for by an allocation instruction; but, by contrast, fails to disclose how many allocation instructions are to be inserted into a routine and/or a function call made from the routine being a stimulus for the insertion of an allocation instruction into the routine.

Because Orr is a wholly irrelevant reference and because Smith fails to disclose any details regarding the insertion of an allocation instruction the combination of Smith and Orr simply fail to cover the matter claimed by independent claim 1 of the present application.

Therefore the Applicants respectfully submit that independent claim 1 and each of its corresponding dependent claims are patentable over the outstanding theories of rejection.

Independent Claim 14

Independent claim 14 presently stands rejected under 35 USC 103 as being obvious in light of the combination of Smith and Orr. Independent claim 14 presently recites:

> 14. A method comprising:
> inserting an on-processor register allocation instruction within a machine code description of a routine because a functional characteristic selected from the group consisting of:
> a) a loop that exists within a control flow graph of said routine;
> b) a software pipelined loop; and,
> c) a function call to another routine;
> is discovered within said routine, said inserted on-processor register allocation instruction to allocate on processor register space for the use of said routine.

The Applicants respectfully submit that Independent claim 14 is patentable over the combination of Orr and Smith because, as discussed above, Orr is a wholly irrelevant reference and Smith fails to disclose any details regarding the insertion of an allocation instruction. In particular, both Orr and Smith fail to disclose any details regarding a particular functional characteristic (e.g., a loop, a software pipelined loop and/or a function call) being a stimulus for the insertion of an allocation instruction. Because neither of these references individually teach or suggest a specific stimulus for the insertion of an allocation instruction, it is

impossible for their combination to disclose, teach or suggest any such stimulus.

Better said, the Applicants claim a causal relationship between the presence of a particular functional characteristic within a routine and the insertion of an allocation instruction within the routine. Because there is simply no relationship between Orr and Smith, the "function call" of Orr cannot be related to (what is at most a suggestion of the existence of) an allocation instruction as provided by Smith.

Therefore independent claim 14 and its corresponding dependent claims are patentable over the combination of Orr and Smith.

Independent Claim 28

Independent claim 28 presently stands rejected under 35 USC 102(b) as being anticipated by Wu. Independent claim 28 presently recites (emphasis added):

> 28. A method, comprising:
> performing a first allocation for a first amount of on-processor register space at the entry block of a routine, said first amount of on-processor register space for the use of said routine;
> performing a second allocation for a second amount of on-processor register space for the storage of live information of said routine when said routine performs a function call to a second routine, said second amount less than said first amount;
> performing said function call to said second routine; and,
> performing a third allocation for a third amount of on-processor register space at the entry block of said second routine, said third amount of allocated on-processor register space for the use of said second routine and including a region of said first amount of allocated on-processor register space that stores stale information of said routine when said routine said performs said function call to said second routine.

The Applicants respectfully submit that independent claim 28 is patentable over Wu for the same reasons discussed above with respect to independent claim 28. That is, claim 28 recites a pair of allocations ("a first allocation" and "a second allocation") that allocate registers for the use of the same routine. Moreover, claim 28 recites that the later (second) allocation allocates for less register space than the earlier allocation.

These claimed features stem at least from the Applicants' discussion of Figures 5 and 6 of the present application. That is, Figures 5 and 6 of the present application and their surrounding discussion disclose a technique where an allocation instruction that allocates register space (e.g., space 610 in Figure 6) for a caller routine (as a consequence of the caller routine having a function call) can be configured to allocate for less register space than a previously executed allocation instruction allocated (e.g., space 605 of Figure 6) for the same caller routine (e.g., located at the caller routine's entry block).

By contrast the Wu reference at best only discloses, teaches or suggests an allocation instruction for the called routine rather than the caller routine. As such, the Wu reference fails to cover the Applicants' claimed subject matter. The Wu reference teaches the implementation of a software pipelined loop as a called sub function and the corresponding allocation of register space for the called sub function. See, Wu Col. 3,

lines 26-29 and Col.4 lines 5-21. Nothing is said in Wu with respect to the insertion of an allocation instruction that allocates register space for the caller function, however. Therefore the Wu reference fails to anticipate each and every limitation of independent claim 28 of the present application.

Therefore independent claim 28 and its corresponding dependent claims are patentable over the Wu reference.

Independent Claims 41, 54, 68

The Applicants respectfully submit that independent claims 41, 54 and 68 and their corresponding dependent claims are patentable for the same reasons put forth with respect to independent claims 1, 14 and 28, respectively.

CONCLUSION

In light of the foregoing comments the Applicants respectfully submit that claims 1 through 80 are patentable and the Applicants respectfully request the allowance of same.

Applicants respectfully submit the present application is in condition for allowance. If the Examiner believes a telephone conference would expedite or assist in the allowance of the present application, the Examiner is invited to call Robert O'Rourke at (408) 720-8300.

Authorization is hereby given to charge our Deposit Account No. 02-2666 for any charges that may be due.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN

Date: 2/19/09

Robert B. O'Rourke
Reg. No. 46,972

12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025-1026
(408) 720-8300